REMARKS

Claims 6-9 have been cancelled. Claims remaining in the present application are Claims 1-5 and 10-18. The drawings have been objected to and have been responded to in a submission of proposed drawing amendments filed herewith. No new matter has been added as a result of these amendments.

DRAWINGS

The drawings were objected to in the first paragraph of the Office Action for having poor figure legends. Formal drawings have been submitted for figures 1-24. No new matter has been added as a result of providing formal drawings.

Double Patenting Rejection

Claims 6 and 7 are provisionally rejected under U.S.C. 101 as claiming the same invention as that of Claims 5 and 6 respectively, of co-pending Application no. 09/867,594. Claims 6-9 have been cancelled herein, thereby obviating the double patenting rejection of Claims 6 and 7.

Claims 1, 8, 10, 15 and 17 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over Claims 1, 7, 9, 13, and 14, respectively, of co-pending US Patent No. 09/867,594. A terminal disclaimer in compliance with 37 CFR § 1.321 is being submitted concurrent with the instant response, thereby obviating the double patenting rejection.

Claim Rejections

35 U.S.C. §102

Claims 1-18 are rejected under U.S.C 102(a) as anticipated by iPlanet Directory Server Administrator's Guide (Version 5.0, April 2001, Sun Microsystems, Inc., hereafter iPlanet).

Applicants respectfully submit that the inventors of the above-mentioned application conceived and invented the subject matter disclosed in above mentioned reference iPlanet. Declarations under 37 CFR §1.132 attesting to this are being filed concurrently with this response. Accordingly, the features of the above-mentioned application as recited in Claims 1-5 and 10-18 are not anticipated by the above mentioned reference iPlanet and the rejection of Claims 1-5 and 10-18 is overcome.

CONCLUSION

In light of the above listed amendments and remarks, reconsideration of the rejected Claims is requested. Based on the amendments and remarks presented above, it is respectfully submitted that Claims 1-5 and 10-18 overcome the rejections of record. Therefore, allowance of Claims 1-5 and 10-18 is earnestly solicited.

Please charge any additional fees or apply any credits to our PTO deposit account number: 23-0085.

Respectfully submitted,

WAGNER, MURABITO & HAO L.L.P.

Dated: 1/27, 2004

Anthony Murabito
Registration No. 35,295

Two North Market Street
Third Floor
San Jose, CA 95113
(408) 938-9060

A

Administrator's Guide

iPlanet Directory Server

Version 5.0

816-0799-01
April 2001

Contents

Introduction **19**
iPlanet Directory Server 5.0 Overview 19
Prerequisite Reading 20
Conventions Used in This Book 21
Related Information 21

Administering iPlanet Directory Server **23**

Chapter 1 Introduction to iPlanet Directory Server **25**
Overview of Directory Server Management 26
Using the Directory Server Console 26
Configuring the Directory Manager 27
Binding to the Directory From iPlanet Console 28
Starting and Stopping the Directory Server 29
Configuring LDAP Parameters 30
Starting the Server with SSL Enabled 33
Cloning a Directory Server 34
Starting the Server in Referral Mode 35

Chapter 2 Creating Directory Entries **37**
Managing Entries From the Directory Console 37
Managing Entries From the Command Line 47
Providing Input From the Command Line 47
Adding and Modifying Entries Using ldapmodify 49
LDIF Update Statements 54
A Note on Renaming Entries 58
Adding Attributes to Existing Entries Using LDIF 59
Deleting an Entry Using LDIF 63
Maintaining Referential Integrity 64
How Referential Integrity Works 64

Using Referential Integrity with Replication 65
Configuring the Supplier Server 65
From the Directory Server Console 66
From the Directory Server Console 67
From the Directory Server Console 68
From the Directory Server Console 68

Chapter 3 Configuring Directory Databases 71
Creating and Maintaining Suffixes 71
Creating Suffixes 72
Maintaining Suffixes 78
Creating and Maintaining Databases 81
Creating Databases 81
Maintaining Directory Databases 86
Creating and Maintaining Database Links 87
Configuring the Chaining Policy 88
Creating a New Database Link 94
Chaining Using SSL 104
Maintaining Database Links 105
Database Links and Access Control Evaluation 106
Advanced Feature: Tuning Database Link Performance 108
Detecting Errors During Normal Processing 111
Managing Threaded Operations 112
Advanced Feature: Configuring Cascading Chaining 113
Overview of Cascading Chaining 113
Summary of Cascading Chaining Configuration Attributes 120
Cascading Chaining Configuration Example 121
Using Referrals 125
Setting Default Referrals 126
Creating Smart Referrals 127
Creating Suffix Referrals 129

Chapter 4 Populating Directory Databases 133
Importing Data 133
Importing From the Command Line 137
Exporting Data 140
Backing Up and Restoring Data 144
Backing Up All Databases 145
Backing Up the dse.ldif Configuration File 147
Restoring All Databases 147
Restoring Databases that Include Replicated Entries 150
Restoring the dse.ldif Configuration File 151

Enabling and Disabling Read-Only Mode 151

Chapter 5 Advanced Entry Management 153
Using Groups 153
Managing Static Groups 154
Managing Dynamic Groups 155
Using Roles 156
About Roles 156
Managing Roles Using the Console 157
Managing Roles Using the Command Line 162
Examples: Managed Role Definition 163
Example: Filtered Role Definition 164
Example: Nested Role Definition 165
Using Roles Securely 165
Assigning Class of Service 166
About CoS 167
About the CoS Definition Entry 167
About the CoS Template Entry 168
How a Pointer CoS Works 169
How an Indirect CoS Works 169
How a Classic CoS Works 170
Managing CoS Using the Console 171
Managing CoS From the Command Line 174
Example of a Pointer CoS 178
Example of an Indirect CoS 179
Example of a Classic CoS 180
Creating Role-Based Attributes 181
Access Control and CoS 182

Chapter 6 Managing Access Control 183
Access Control Principles 184
ACI Structure 184
ACI Placement 185
ACI Evaluation 185
ACI Limitations 186
Default ACIs 187
Creating ACIs Manually 188
The ACI Syntax 188
Example ACI 189
Targeting Attributes 192
Rights Required for LDAP Operations 198
Permissions Syntax 199

Bind Rules 200
Bind Rule Syntax 200
Anonymous Access (anyone Keyword) 202
General Access (all Keyword) 203
Self Access (self Keyword) 203
Parent Access (parent Keyword) 203
LDAP URLs 203
Wildcards 204
Examples 204
Examples 206
Examples 215
Examples 216
Creating ACIs From the Console 218
Access Control Usage Examples 224
Granting Anonymous Access 225
Granting Write Access to Personal Entries 227
Restricting Access to Key Roles 231
Granting a Group Full Access to a Suffix 232
Granting Rights to Add and Delete Group Entries 233
Granting Conditional Access to a Group or Role 235
Denying Access 238
Setting a Target Using Filtering 240
Allowing Users to Add or Remove Themselves From a Group 240
Defining Permissions for DNs That Contain a Comma 242
Proxied Authorization ACI Example 242
Viewing the ACIs for an Entry 243
Advanced Access Control: Using Macro ACIs 244
Macro ACI Example 244
Macro ACI Syntax 247
Macro Matching for ($dn) 248
Macro Matching for [$dn] 248
Macro Matching for ($attr.*attrName*) 249
Access Control and Replication 250
Logging Access Control Information 251
Compatibility with Earlier Releases 251

Chapter 7 User Account Management 253
Managing the Password Policy 253
Configuring the Password Policy 254
Configuring the Password Policy Using the Console 254
Configuring the Password Policy Using the Command-Line 255
Setting User Passwords 259
Configuring the Account Lockout Policy 260

Configuring the Account Lockout Policy Using the Console ... 260
Configuring the Account Lockout Policy Using the Command Line ... 260
Managing the Password Policy in a Replicated Environment ... 262
Inactivating Users and Roles ... 263
Inactivating User and Roles Using the Console ... 264
Inactivating User and Roles Using the Command Line ... 264
Activating User and Roles Using the Console ... 265
Activating User and Roles Using the Command Line ... 266
Setting Resource Limits Based on the Bind DN ... 266
Setting Resource Limits Using the Console ... 267
Setting Resource Limits Using the Command Line ... 267

Chapter 8 Managing Replication ... 269
Replication Overview ... 270
Read-Write Replica/Read-Only Replica ... 270
Supplier/Consumer ... 270
Change Log ... 271
Unit of Replication ... 271
Replication Identity ... 271
Replication Agreement ... 272
Compatibility with Earlier Versions of Directory Server ... 272
Replication Scenarios ... 273
Single-Master Replication ... 273
Multi-Master Replication ... 274
Cascading Replication ... 276
Configuring Single-Master Replication ... 277
Configuring Multiple-Master Replication ... 280
Configuring Cascading Replication ... 285
Configuration Tips ... 290
Detailed Procedures ... 291
Removing the Change Log ... 296
Initializing Consumers ... 297
When to Initialize a Consumer ... 297
Exporting a Replica to LDIF ... 300
Importing the LDIF File to the Consumer Server ... 300
Forcing Replication Updates ... 300
Replication over SSL ... 304
Replication with Earlier Releases ... 306
Using the Retro Change Log Plug-In ... 308
Monitoring Replication Status ... 311
Solving Common Replication Conflicts ... 312
Naming Conflicts ... 313
Deleted Entries with Child Entries ... 315

Controlling Access to Conflicting Entries 315

Chapter 9 Extending the Directory Schema 317
Overview of Extending Schema 317
Managing Attributes 318
Managing Object Classes 321
Turning Schema Checking On and Off 326

Chapter 10 Managing Indexes 327
About Indexes 327
About Index Types 328
About Default, System, and Standard Indexes 329
Overview of Default Indexes 329
Overview of System Indexes 331
Overview of Standard Indexes 331
Overview of the Searching Algorithm 332
Balancing the Benefits of Indexing 334
Creating Indexes 336
Creating Indexes From the Server Console 337
Creating Indexes From the Command Line 338
Adding an Index Entry 338
Running the db2index.pl Script 340
The following table describes the `db2index.pl` options used in the examples: 341
Creating Browsing Indexes From the Server Console 341
Creating Browsing Indexes from the Command Line 342
Adding a Browsing Index Entry 342
Running the vlvindex Script 344
Deleting Indexes 345
Deleting Indexes From the Server Console 346
Deleting Indexes From the Command Line 347
Deleting an Index Entry 347
Running the db2index.pl Script 348
Deleting Browsing Indexes From the Server Console 349
Deleting Browsing Indexes From the Command Line 350
Deleting a Browsing Index Entry 350
Running the vlvindex Script 352
Managing Indexes 353
Benefits of the All IDs Mechanism 353
Drawbacks of the All IDs Mechanism 354
When All IDs Threshold is Too Low 354
When All IDs Threshold is Too High 355
All IDs Threshold Tuning Advice for Single- Enterprise Directories 355

All IDs Threshold Tuning Advice for Service Providers and Extranets 356
Default All IDs Threshold Value 357
Symptoms of an Inappropriate All IDs Threshold Value 357
Changing the All IDs Threshold Value 358
Attribute Name Quick Reference Table 359

Chapter 11 Managing SSL 361
Introduction to SSL in the Directory Server 361
Obtaining and Installing Server Certificates 363
Activating SSL 368
Setting Security Preferences 369
Using Certificate-Based Authentication 371
Configuring LDAP Clients to Use SSL 373

Chapter 12 Monitoring Server and Database Activity 377
Viewing and Configuring Log Files 377
Access Log 379
Error Log 380
Audit Log 382
Manual Log File Rotation 384
Monitoring Server Activity 384
Viewing the Server Performance Monitor 385
Overview of Server Performance Monitor Information 385
General Information (Server) 385
Resource Summary 386
Current Resource Usage 387
Connection Status 387
Global Database Cache Information 388
Monitoring Database Activity 391
Viewing Database Performance Monitors 391
Overview of Database Performance Monitor Information 392
General Information (Database) 392
Summary Information Table 392
Database Cache Information Table 393
Database File-Specific Table 394
Monitoring Database Link Activity 397

Chapter 13 Monitoring Directory Server Using SNMP 399
About SNMP 399
SNMP Overview 400
NMS-Initiated Communication 401
Managed Device-Initiated Communication 401

Overview of the Directory Server Management Information Base ... 402
About the Operations Table ... 402
The Entries Table ... 404
Setting Up SNMP ... 405
Setting Up SNMP on Windows NT ... 405
Setting Up SNMP on UNIX ... 405
Configuring the AIX SNMP Daemon ... 406
Starting and Stopping the SNMP Subagent on UNIX ... 407
Starting and Stopping the SNMP Service on Windows NT ... 407
Configuring SNMP for the Directory Server ... 408

Chapter 14 Tuning Directory Server Performance ... 409
Tuning Server Performance ... 409
Tuning Database Performance ... 410
Optimizing Search Performance ... 410
Tuning Transaction Logging ... 413
Changing the Location of the Database Transaction Log ... 414
Changing the Database Checkpoint Interval ... 414
Disabling Durable Transactions ... 415

iPlanet Plug-Ins Reference ... 417

Chapter 15 Administering Directory Server Plug-Ins ... 419
Server Plug-in Functionality Reference ... 419
7-bit Check Plug-In ... 420
ACL Plug-In ... 421
ACL Preoperation Plug-In ... 421
Binary Syntax Plug-In ... 422
Boolean Syntax Plug-In ... 422
Case Exact String Syntax Plug-In ... 423
Case Ignore String Syntax Plug-In ... 423
Chaining Database Plug-In ... 424
Class of Service Plug-In ... 424
Country String Syntax Plug-In ... 425
Distinguished Name Syntax Plug-In ... 425
Generalized Time Syntax Plug-In ... 426
Integer Syntax Plug-In ... 427
Internationalization Plug-In ... 427
ldbm Database Plug-In ... 428
Legacy Replication Plug-In ... 428
Multimaster Replication Plug-In ... 429
Octet String Syntax Plug-in ... 429
CLEAR Password Storage Plug-In ... 430

CRYPT Password Storage Plug-In 430
NS-MTA-MD5 Password Storage Plug-In 431
SHA Password Storage Plug-In 432
SSHA Password Storage Plug-in 432
Postal Address String Syntax Plug-In 433
PTA Plug-In 433
Referential Integrity Postoperation Plug-In 434
Retro Change Log Plug-In 435
Roles Plug-In 435
Telephone Syntax Plug-In 436
UID Uniqueness Plug-in 437
URI Plug-in 439
Enabling and Disabling Plug-Ins From the Server Console 439

Chapter 16 Using the Pass-Through Authentication Plug-In 441
How Directory Server 5.0 Uses PTA 441
PTA Plug-In Syntax 443
Configuring the PTA Plug-In 445
PTA Plug-In Syntax Examples 451

Chapter 17 Using the Attribute Uniqueness Plug-In 455
Overview of the Attribute Uniqueness Plug-In 455
Overview of the UID Uniqueness Plug-in 457
Attribute Uniqueness Plug-In Syntax 457
Creating an Instance of the Attribute Uniqueness Plug-In 460
Configuring Attribute Uniqueness Plug-Ins 461
Configuring Attribute Uniqueness Plug-Ins From the Directory Server Console 461
Attribute Uniqueness Plug-In Syntax Examples 465
Replication and the Attribute Uniqueness Plug-In 467
Simple Replication Scenario 467
Multi-Master Replication Scenario 467

Appendixes 469

Appendix A LDAP Data Interchange Format 471
LDIF File Format 471
Continuing Lines in LDIF 473
Representing Binary Data 473
Specifying Directory Entries Using LDIF 475
Defining Directories Using LDIF 480
LDIF File Example 481
Storing Information in Multiple Languages 482

Appendix B Finding Directory Entries **485**
Finding Entries Using the Server Console 485
Using ldapsearch 486
Using Special Characters 486
ldapsearch Command-Line Format 487
Commonly Used ldapsearch options 487
ldapsearch Examples 489
Returning All Entries 490
Specifying Search Filters on the Command Line 490
Searching the Root DSE Entry 490
Searching the Schema Entry 490
Using LDAP_BASEDN 491
Displaying Subsets of Attributes 491
Specifying Search Filters Using a File 491
Specifying DNs that Contain Commas in Search Filters 492
Using Client Authentication When Searching 492
LDAP Search Filters 493
Search Filter Syntax 493
Using Attributes in Search Filters 494
Using Operators in Search Filters 494
Using Compound Search Filters 496
Search Filter Examples 496
Searching an Internationalized Directory 497
Matching Rule Filter Syntax 498
Matching Rule Formats 498
Using Wildcards in Matching Rule Filters 500
Supported Search Types 501
International Search Examples 502
Less Than Example 502
Less Than or Equal to Example 502
Equality Example 503
Greater Than or Equal to Example 503
Greater Than Example 503
Substring Example 503

Appendix C LDAP URLs **505**
Components of an LDAP URL 505
Escaping Unsafe Characters 507
Examples of LDAP URLs 508

Appendix D Internationalization **511**
About Locales 511
Identifying Supported Locales 513

Supported Language Subtypes 515

Glossary **517**

Index **533**

List of Figures

Figure 1-1 Viewing the Bind DN 29
Figure 2-1 Directory Server Console - Property Editor 42
Figure 6-1 Using Inheritance With the `userattr` Keyword 211
Figure 6-2 Selecting an Object in the Navigation Tree to Set Access Control 220
Figure 6-3 Access Control Editor Window 221
Figure 6-4 Example directory tree for Macro ACIs 245
Figure 8-1 Single-Master Replication 274
Figure 8-2 Multi-Master Replication 275
Figure 8-3 Cascading Replication 276

List of Tables

Table 2-1 Entry Templates and Corresponding Object Classes 39
Table 3-1 Suffix Attributes 76
Table 3-2 Components Allowed to Chain 89
Table 3-3 LDAP Controls and Their OIDs 93
Table 3-4 Database Link Configuration Attributes 100
Table 3-5 Database Link Connection Management Attributes 110
Table 3-6 Database Link Processing Error Detection Parameters 111
Table 3-7 Cascading Chaining Configuration Attributes 120
Table 4-1 Import Method Comparison 134
Table 5-1 CoS Definition Entry Object Classes 174
Table 5-2 CoS Definition Entry Attributes 175
Table 5-3 CoS Definitions 176
Table 6-1 LDIF Target Keywords 190
Table 6-2 LDIF Bind Rule Keywords 201
Table 6-3 Macros in ACI Keywords 247
Table 7-1 Password Policy Attributes 256
Table 7-2 Account Lockout Policy Attributes 261
Table 8-1 Replicate_Now Variables 304
Table 8-2 Attributes of a Retro Change Log Entry 309
Table 8-3 Directory Server Console - Status Tab 312
Table 9-1 Attributes Tab Reference 318
Table 9-2 Object Classes Tab Reference 322
Table 10-1 Default indexes 330
Table 10-2 System indexes 331
Table 10-3 Attribute Name Quick Reference Table 359
Table 12-1 Server Performance Monitoring - Resource Summary Table 386
Table 12-2 Server Performance Monitoring - Current Resource Usage Table 387

Table 12-3 Server Performance Monitoring - Connection Status Table 388
Table 12-4 Server Performance Monitoring - Global Database Cache Table 388
Table 12-5 Database Performance Monitoring - Summary Information 392
Table 12-6 Database Performance Monitoring - Database Cache Information 393
Table 12-7 Database Performance Monitoring - Database File-Specific table 394
Table 12-8 Database Link Monitoring Attributes 397
Table 13-1 Operations Table Managed Objects and Descriptions 403
Table 13-2 Entries Table Managed Objects and Descriptions 404
Table 16-1 PTA Plug-In Parameters 444
Table 17-1 Attribute Uniqueness Plug-In Variables 459
Table A-1 LDIF Fields 472
Table A-2 LDIF Elements in Organization Entries 476
Table A-3 LDIF Elements in Organizational Unit Entries 477
Table A-4 LDIF Elements in Person Entries 479
Table B-1 Search Filter Operators 494
Table B-2 Search Filter Boolean Operators 496
Table B-3 Search Types, Operators, and Suffixes 501
Table C-1 LDAP URL Components 505
Table D-1 Supported Locales 513
Table D-2 Supported Language Subtypes 515

2. Double-click the entry you want to modify or select Open from the Object menu.

   The Edit Group dialog box appears.

3. Make your changes to the group information. Click OK.

   To view your changes, go to the View menu and select Refresh.

Using Roles

Roles are a new entry grouping mechanism that unify the static and dynamic groups described in the previous sections. Roles are designed to be more efficient and easier to use for applications. For example, an application can locate the role of an entry, rather than select a group and browse the members list.

This section contains the following topics:

- "About Roles," on page 156
- "Managing Roles Using the Console," on page 157
- "Managing Roles Using the Command Line," on page 162
- "Using Roles Securely," on page 165

About Roles

Roles unify the static and dynamic group concept supported by previous versions of Directory Server.

You can use roles to:

- Enumerate the members of a role.

  Having an enumerated list of role members can be useful for resolving queries for role members quickly.

- Determine whether a given entry possesses a particular role.

  Knowing the roles possessed by an entry can help you determine whether the entry possesses the target role.

- Enumerate all the roles possessed by a given entry.
- Assign a particular role to a given entry.

- Remove a particular role from a given entry.

You can do everything you would normally do with static groups with managed roles, and you can filter members using filtered roles as you used to do with dynamic groups. Roles are easier to use than groups, more flexible in their implementation, and reduce client complexity.

However, evaluating roles is more resource intensive because the server does the work for the client application. With roles, the client application can check role membership by searching the `nsRole` attribute. The `nsRole` attribute is a computed attributed that is not stored with the entry itself, which identifies which roles an entry belongs to. From the client application point of view, the method for checking membership is uniform and is performed on the server side.

Each role has *members*, or entries that possess the role. You can specify members either explicitly or dynamically. How you specify role membership depends upon the type of role you are using. Directory Server supports three types of roles:

- Managed roles.

  A managed role allows you to create an explicit enumerated list of members.

- Filtered roles.

  A filtered role allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

- Nested roles.

  A nested role allows you to create roles that contain other roles.

For more information about how roles work, refer to *iPlanet Directory Server Deployment Guide*.

Managing Roles Using the Console

This section contains the following procedures for creating and modifying roles:

- "Creating a Managed Role," on page 158
- "Creating a Filtered Role," on page 159
- "Creating a Nested Role," on page 160
- "Viewing and Editing an Entry's Roles," on page 160
- "Modifying a Role Entry," on page 161

- "Making a Role Inactive," on page 161
- "Reactivating a Role," on page 162
- "Deleting a Role," on page 162

When you create a role, you need to decide whether a user can add themselves or remove themselves from the role. Refer to "Using Roles Securely," on page 165 for more information about roles and access control.

Creating a Managed Role

Managed roles allow you to create an explicit enumerated list of members. Managed roles are added to entries by adding the `nsRoleDN` attribute to the entry.

To create and add members to a managed role:

1. On the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry for your new role.
3. Go to the Object menu and select New > Role.

   You can also right click the entry and select New > Role.

   The Create New Role dialog box is displayed.
4. Click General in the left pane. Type a name for your new role in the "Role Name" field.

   The role name is required.
5. Enter a description of the new role in the "Description" field.
6. Click Members in the left pane.

   A search dialog box appears briefly.
7. In the right pane, select Managed Role. Click Add to add new entries to the list of members.

   The standard "Search users and groups" dialog box appears.
8. In the Search drop-down list, select Users from the Search drop-down list, then click Search. Select one of the entries returned and click OK.
9. When you have finished adding entries to the role, click OK.

   The new role appears in the right pane.

Creating a Filtered Role

You assign entries to a filtered role depending upon a particular attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

To create and add members to a filtered role:

1. Follow steps 1-5 of "Creating a Managed Role," on page 158.
2. Click Members in the left pane.

   A search dialog box appears briefly.
3. In the right pane, select Filtered Role.
4. Enter an LDAP filter in the text field, or click Construct to be guided through the construction of an LDAP filter.
5. If you click Construct, the standard LDAP URL construction dialog appears. Disregard the fields identifying the host server, port number and base DN.

   c. From the "Search" drop-down list, select whether the filter searches for entries one level below the base DN, or searches the whole subtree beneath the base DN. Do not use the base DN only option, because roles are designed to work only on entries within their scope.

   d. Select the types of entries you want to filter from the "For" drop-down list.

      You can choose between users, groups, or both.

   e. Select an attribute from the "Where" drop-down list. The two fields following it allow you to refine your search by selecting one of the qualifiers from the drop-down list (such as contains, does not contain, is, is not) and enter an attribute value in the text box. To add additional filters, click More. To remove unnecessary filters, click Fewer.

   f. Click OK to save your filter.
6. Click Test to try your filter.

   A Filter Test Result dialog box displays the entries matching your filter.
7. Click OK.

   The new role appears in the right pane.

Creating a Nested Role

Nested roles allow you to create roles that contain other roles. Before you create a nested role, another role must exist. When you create a nested role, the console displays a list of the roles available for nesting. The roles nested within the nested role are specified using the `nsRoleDN` attribute.

To create and add members to a nested role:

1. Follow steps 1-5 of "Creating a Managed Role," on page 158.
2. Click Members in the left pane.

   A search dialog box appears briefly.
3. In the right pane, select Nested Role.
4. Click Add to add roles to the list.The members of the nested role are members of other existing roles.

   The Role Selector dialog box appears.
5. Select a role from the "Available roles" list and click OK.
6. Click OK.

   The new role appears in the right pane.

Viewing and Editing an Entry's Roles

To view or edit a role associated with an entry from the console:

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the entry for which you want to view or edit a role. Select Set Roles from the Object menu.

   The Roles dialog box displays.
3. Select the Managed Roles tab to display the managed roles to which this entry belongs.
4. To add a new managed role, click Add and select an available role from the Role Selector window. Click OK.

   To remove a managed role, select it and click Remove.

   To edit a managed role associated with an entry, click Edit. The Edit Entry dialog box displays. Make any changes to the general information or members and click OK.

5. Select the Other Roles tab to view what filtered or nested roles this entry belongs to.
6. Click Edit to make changes to any filtered or nested roles associated with the entry. Click OK to save your changes.
7. Click OK once you have finished modifying the roles to save your changes.

Modifying a Role Entry

To edit an existing role:

1. On the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Double-click the role.

   The Edit Entry dialog box appears.
4. Click General in the left pane to change the role name and description.
5. Click Members in the left pane to change the members of managed and nested roles or to change the filter of a filtered role.
6. Click OK to save your changes.

Making a Role Inactive

You can temporarily disable the members of a role by inactivating the role to which they belong. Inactivating a role inactivates the entries possessed by the role and not the role itself.

To temporarily disable the members of a role:

1. On the Directory Server Console, select the Directory tab.
2. Browse the navigation tree in the left pane to locate the base DN for your role. Roles appear in the right pane with other entries.
3. Select the role. Select Inactivate from the Object menu.

   You can also right-click the role and select Inactivate from the menu.

   The role is inactivated.

   To see the inactivated entries, select Inactivation State from the View menu. A red slash through the role icon indicates that the role has been inactivated.

To prevent users from removing the `nsRoleDN` attribute, use the following ACIs depending upon the type of role being used.

Managed roles. For entries that are members of a managed role, use the following ACI to prevent users from unlocking themselves by removing the appropriate `nsRoleDN`:

```
aci: (targetattr="nsRoleDN")
     (targattrfilters="

  add=nsRoleDN:(!(nsRoleDN=cn=AdministratorRole,dc=siroe,dc=com)),

  del=nsRoleDN:(!(nsRoleDN=cn=nsManagedDisabledRole,dc=siroe,dc=com
))")
(version3.0;aci "allow mod of nsRoleDN by self
     but not to critical values";
     allow(write)
     userdn="ldap:///self";)
```

Filtered roles. The attributes that are part of the filter should be protected so that the user cannot relinquish the filtered role by modifying an attribute. The user should not be allowed to add, delete, and modify the attribute used by the filtered role. If the value of the filter attribute is computed, then all attributes that can modify the value of the filter attribute should be protected in the same way.

Nested roles. A nested role is comprised of filtered and managed roles, so the above points should be considered for each of the roles that comprise the nested role.

For more information about account inactivation, see "Inactivating Users and Roles," on page 263.

Assigning Class of Service

A class of service (CoS) allows you to share attributes between entries in a way that is transparent to applications. CoS simplifies entry management and reduces storage requirements.

There are two methods for creating and managing CoS, using the Directory Server console or through the command line. The following sections describe CoS in more detail and provide the procedures for managing CoS through both the console and the command line:

- "About CoS," on page 167
- "Managing CoS Using the Console," on page 171

- "Managing CoS From the Command Line," on page 174
- "Creating Role-Based Attributes," on page 181
- "Access Control and CoS," on page 182

About CoS

Clients of the Directory Server read the attributes on a user's entry. With CoS, some attribute values may not be stored with the entry itself. Instead, they are generated by class of service logic as the entry is sent to the client application.

Each CoS is comprised of the following two types of entry in your directory:

- CoS Definition Entry.

  The CoS definition entry identifies the type of CoS you are using. Like the role definition entry, it inherits from the `LDAPsubentry` object class. The CoS definition entry is below the branch at which it is effective.

- Template Entry.

  The CoS template entry contains a list of the shared attribute values. Changes to the template entry attribute values are automatically applied to all the entries within the scope of the CoS. A single CoS might have more than one template entry associated with it.

The CoS definition entry and template entry interact to provide attribute information to their ***target entries***, any entry within the scope of the CoS.

NOTE LDAP search requests containing a filter that references an attribute defined by a CoS may return unexpected results. Take care when deciding which attributes to generate using a CoS.

The following sections describe the entries that make up a CoS in more detail and provide examples of each type of CoS.

About the CoS Definition Entry

The CoS definition entry is an instance of the `cosSuperDefinition` object class. The CoS definition entry also contains an object class that specifies the type of template entry it uses to generate the entry. You can specify three different object classes depending upon the type of CoS you want to use. The target entries share the same parent as the CoS definition entry.

There are 3 types of CoS, defined using three types of CoS definition entries:

- Pointer CoS.

  A pointer CoS identifies the template entry using the template DN only.

- Indirect CoS.

  An indirect CoS identifies the template entry using the value of one of the target entry's attributes. For example, an indirect CoS might specify the `manager` attribute of a target entry. The value of the `manager` attribute is then used to identify the template entry.

  The target entry's attribute must be single-valued and contain a DN.

- Classic CoS.

  A classic CoS identifies the template entry using a combination of the template entry's base DN and the value of one of the target entry's attributes.

For more information about the object classes and attributes associated with each type of CoS, refer to "Managing CoS From the Command Line," on page 174.

If the CoS logic detects that an entry contains an attribute for which the CoS is generating values, by default the CoS supplies the client application with the attribute value in the entry itself. However, you can use the CoS definition entry to control this behavior.

About the CoS Template Entry

The CoS template entry contains the value or values of the attributes generated by the CoS logic. The CoS template entry contains a general object class of `cosTemplate`. The CoS template entries for a given CoS are stored in the directory tree along with the CoS definition.

The relative distinguished name (RDN) of the template entry is determined by one of the following:

- The DN of the template entry alone.

  This type of template is associated with a pointer CoS.

- The value of one of the target entry's attributes.

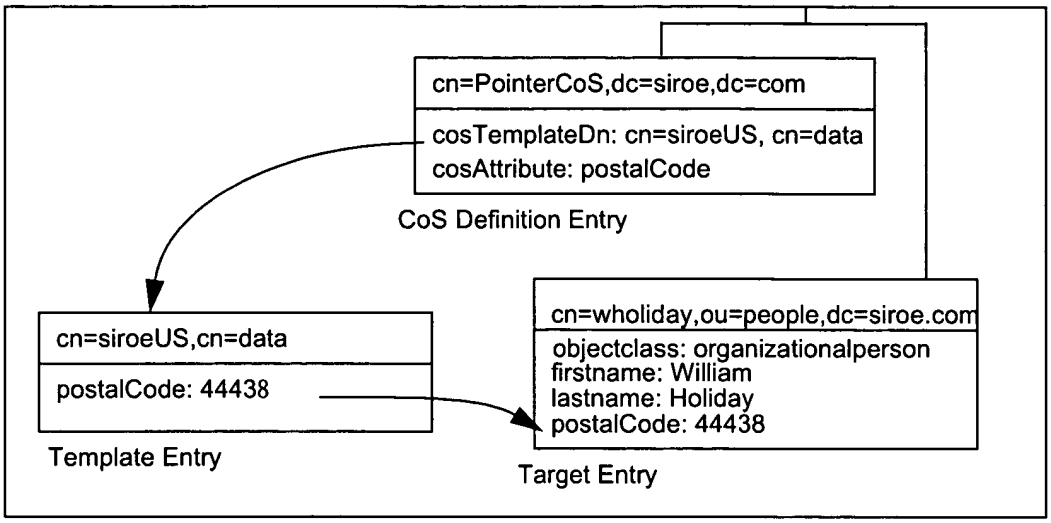
  The attribute used to provide the relative DN to the template entry is specified in the CoS definition entry using the `cosIndirectSpecifier` attribute. This type of template is associated with an indirect CoS.

- By a combination of the DN of the subtree where the CoS performs a one level search for templates and the value of one of the target entry's attributes.

This type of template is associated with a classic CoS.

How a Pointer CoS Works

You create a CoS that shares a common postal code with all of the entries stored under `dc=siroe,dc=com`. The three entries for this CoS appear as follows:

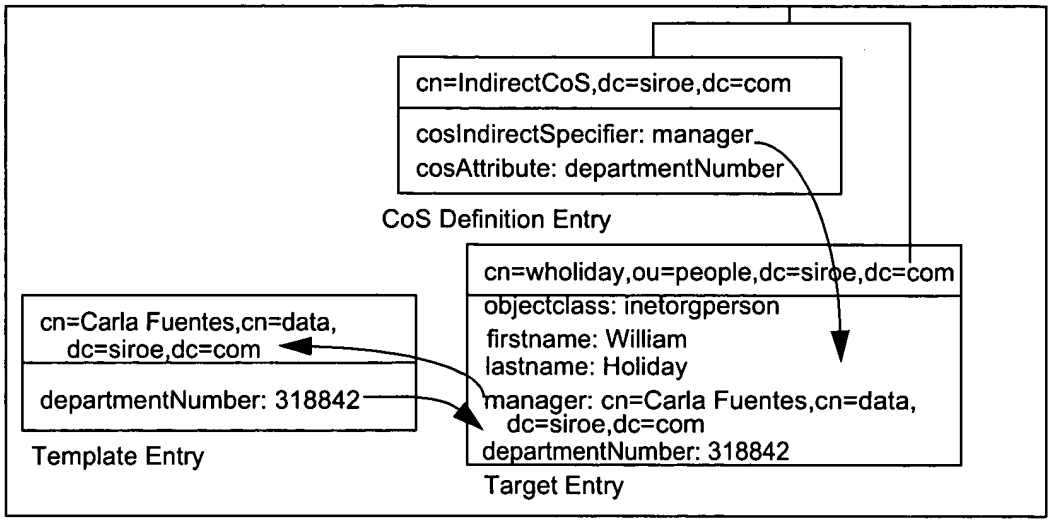


In this example, the template entry is identified by its DN, `cn=siroeUS,cn=data`, in the CoS definition entry. Each time the `postalCode` attribute is queried on the entry `cn=wholiday,ou=people,dc=siroe,dc=com`, the Directory Server returns the value available in the template entry `cn=siroeUS,cn=data`.

How an Indirect CoS Works

You can create an indirect CoS that uses the `manager` attribute of the target entry to identify the template entry.

The three CoS entries appear as follows:

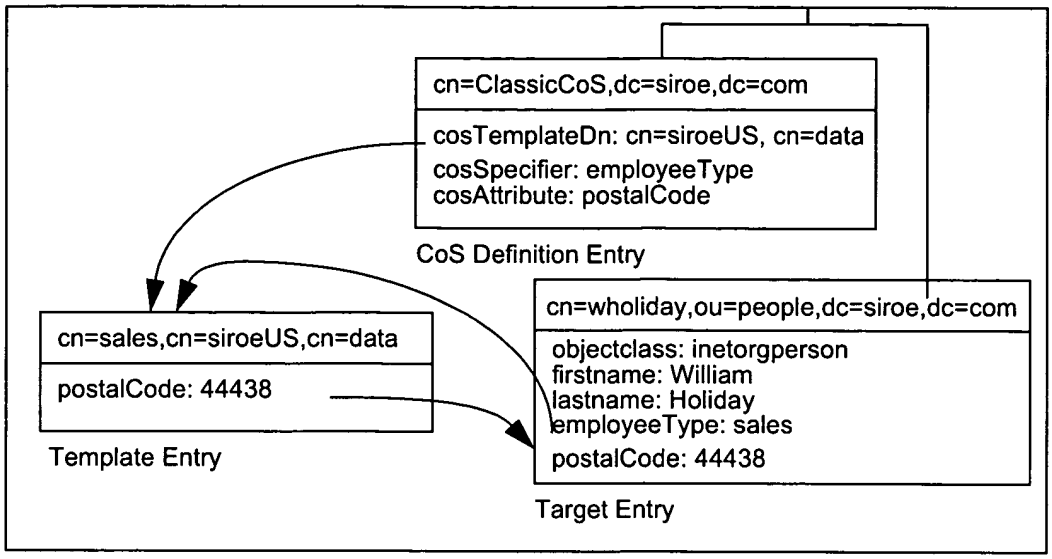


In this example, the target entry for William Holiday contains the indirect specifier, the `manager` attribute. William's manager is Carla Fuentes, so the `manager` attribute contains a pointer to the DN of the template entry, `cn=Carla Fuentes,ou=people,dc=siroe,dc=com`. The template entry in turn provides the `departmentNumber` attribute value of 318842.

How a Classic CoS Works

You can create a classic CoS that uses a combination of the template DN and a CoS specifier to identify the template entry containing the postal code.

The three CoS entries appear as follows:



In this example, the Cos definition entry's `cosSpecifier` attribute specifies the `employeeType` attribute. This attribute, in combination with the template DN, identify the template entry as `cn=sales, cn=siroeUS, cn=data`. The template entry then provides the value of the `postalCode` attribute to the target entry.

Managing CoS Using the Console

This section describes creating and editing CoS through the Directory Server Console. It includes the following sections:

- "Creating a New CoS," on page 171
- "Editing an Existing CoS," on page 173
- "Deleting a CoS," on page 173

Creating a New CoS

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry for your new class of service.
3. Go to the Object menu and select New > Class of Service.

   You can also right click the entry and select New > Class of Service.

   The Create New Class of Service dialog displays.

4. Select General in the left pane. In the right pane, enter the name of your new class of service in the "Class Name" field. Enter a description of the class in the "Description" field.

5. Click Attributes in the left pane. The right pane displays a list of attributes generated on the target entries.

   Click Add to browse the list of possible attributes and add them to the list.

6. Once you have added an attribute to the list, a drop-down list appears in the "Class of Service Behavior" column.

   Select "Does not override target entry attribute" to tell the directory to only return a generated value if there is no corresponding attribute value stored with the entry.

   Select "Overrides target entry attribute" to make the value of the attribute generated by the CoS override the local value.

   Select "Overrides target entry attribute and is operational" to make the attribute override the local value and to make the attribute operational, so that it is not visible to client applications unless explicitly requested.

NOTE You can only make an attribute operational if it is also defined as operational in the schema.

For example, if your CoS generates a value for the `description` attribute, you cannot select "Overrides target entry attribute and is operational" because this attribute is not marked operational in the schema.

---

7. Click Template in the left pane. In the right pane, select how the template entry is identified.

   By its DN. If you choose to have the template entry identified by only its DN (a pointer CoS), enter the DN of the template in the "Template DN" field. Click Browse to locate the DN on your local server.

   Using the value of one of the target entry's attribute. If you choose to have the template entry identified by the value of one of the target entry's attributes (an indirect CoS), enter the attribute name in the "Attribute Name" field. Be sure to select an attribute which contains DN values. Click Change to select a different attribute from the list of available attributes.

Using both its DN and the value of one of the target entry's attributes. If you choose to have the template entry identified by both its DN and the value of one of the target entry's attributes (a classic CoS), enter both a template DN and an attribute name.

8. Click OK.

Editing an Existing CoS

The following procedure describes changing the description and attributes generated on the target entry of an existing class of service.

To edit an existing CoS:

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry that contains your class of service.

   The CoS appears in the right pane with other entries.
3. Double-click the CoS.

   The Edit Entry dialog box appears.
4. Click General in the left pane to change the CoS name and description.
5. Click Attributes in the left pane to add or remove attributes generated by the CoS.
6. Click OK to save your changes.

   The target entries of the CoS are automatically updated.

Deleting a CoS

The following procedure describes deleting a CoS:

1. In the Directory Server Console, select the Directory tab.
2. Browse the tree in the left navigation pane and select the parent entry that contains your class of service.

   The CoS appears in the right pane with other entries.
3. Right-click the CoS and select Delete. A dialog box appears asking you to confirm the deletion. Click Yes.
4. The Deleted Entries dialog box appears to inform you that the CoS was successfully deleted. Click OK.

Managing CoS From the Command Line

Because all configuration information and template data is stored as entries in the directory, you can use standard LDAP tools for CoS configuration and management. This section contains the following topics:

- "Creating the CoS Definition Entry From the Command Line," on page 174
- "Creating the CoS Template Entry From the Command Line," on page 177
- "Example of a Pointer CoS," on page 178
- "Example of an Indirect CoS," on page 179
- "Example of a Classic CoS," on page 180

Creating the CoS Definition Entry From the Command Line

Each type of CoS requires a particular object class to be specified in the definition entry. All CoS definition object classes inherit from the `LDAPsubentry` object class and the `cosSuperDefinition` object class. The following table lists the object classes associated with each type of CoS definition entry:

Table 5-1 CoS Definition Entry Object Classes

CoS Type	Object Classes	Description
`Pointer CoS`	`cosPointerDefinition`	Identifies the template entry associated with the CoS definition using the template entry's DN value. The DN of the template entry is specified in the `cosTemplateDn` attribute.
`Indirect CoS`	`cosIndirectDefinition`	Identifies the template entry using the value of one of the target entry's attributes. The attribute of the target entry is specified in the `cosIndirectSpecifier` attribute.
`Classic CoS`	`cosClassicDefinition`	Identifies the template entry using both the template entry's DN (as specified in the `cosTemplateDn` attribute) and the value of one of the target entry's attributes (as specified in the `cosSpecifier` attribute).

You can use the following attributes in your CoS definition entries:

Table 5-2 CoS Definition Entry Attributes

Attribute	Definition
`cosAttribute`	Provides the name of the attribute for which you want to generate a value. You can specify more than one `cosAttribute` value. This attribute is used by all types of CoS definition entries.
`cosIndirectSpecifier`	Specifies the attribute value used by an indirect CoS to identify the template entry.
`cosSpecifier`	Specifies the attribute value used by a classic CoS, which, along with the template entry's DN, identifies the template entry.
`cosTemplateDn`	Provides the DN of the template entry associated with the CoS definition. Used for pointer CoS and classic CoS only.

The `cosAttribute` attribute allows an additional qualifier after the attribute value. You can use the following qualifiers:

- Default

  This qualifier indicates that the server only returns a generated value if there is no corresponding attribute value stored with the entry.

- Override

  This qualifier indicates that the server always returns the value generated by the CoS, even when there is a value stored with the entry.

- Operational

  This qualifier indicates that the attribute will only be returned if it is explicitly requested in the search. Operational attributes do not need to pass a schema check in order to be returned. When you use `operational` as a qualifier, it works as if `override` and `operational` were specified.

NOTE You can only make an attribute operational if it is also defined as operational in the schema.

For example, if your CoS generates a value for the `description` attribute, you use the `operational` qualifier because this attribute is not marked operational in the schema.

If you do not indicate a qualifier, `default` is assumed.

For example, you might create a pointer CoS definition entry that contains an `override` qualifier as follows:

```
dn: cn=pointerCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=siroeUS,cn=data
cosAttribute: postalCode override
```

This pointer CoS definition entry indicates that it is associated with a template entry, `cn=siroeUS,cn=data`, that generates the value of the `postalCode` attribute. The override qualifier indicates that this value will take precedence over the value stored by the entries for the `postalCode` attribute.

NOTE If an entry contains an attribute value generated by a CoS, you cannot manually update the value of the attribute if it is defined with the operational or override qualifiers.

For more information about the attributes, refer to the *iPlanet Directory Server Configuration, Command, and File Reference.*

Now that you have been introduced to the object classes and attributes used by a CoS definition, it is time to put them together to create the definition entry itself. The following table describes the CoS definition for each type of CoS:

Table 5-3 CoS Definitions

CoS Type	CoS definition

Pointer CoS	`objectclass: top`
	`objectclass: LDAPsubentry`
	`objectclass: cosSuperDefinition`
	`objectclass: cosPointerDefinition`
	`cosTemplateDn: DN_string`
	`cosAttribute: list_of_attributes qualifier`

Indirect CoS	`objectclass: top`
	`objectclass: LDAPsubentry`
	`objectclass: cosSuperDefinition`
	`objectclass: cosIndirectDefinition`
	`cosIndirectSpecifier: attribute_name`
	`cosAttribute: list_of_attributes qualifier`

Table 5-3 CoS Definitions

CoS Type	CoS definition

Classic CoS	`objectclass: top`
	`bbjectclass: LDAPsubentry`
	`objectclass: cosSuperDefinition`
	`objectclass: cosClassicDefinition`
	`cosTemplateDn: DN_string`
	`cosSpecifier: attribute_name`
	`cosAttribute: list_of_attributes qualifier`

Creating the CoS Template Entry From the Command Line

The CoS template entry also inherits from the `LDAPsubentry` object class. Each template entry is an instance of the `cosTemplate` object class.

NOTE Making the CoS template entry an instance of the `LDAPsubentry` object classes allows ordinary searches to be performed unhindered by the configuration entries. However, if the template entry already exists and is used for something else (for example, if it is a user entry), you do not need to make it an instance of the `LDAPsubentry` object class.

The CoS template entry also contains the attribute generated by the CoS (as specified in the `cosAttribute` attribute of the CoS definition entry) and the value for that attribute.

For example, a CoS template entry that provides a value for the `postalCode` attribute follows:

```
dn:cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438
```

It is possible to create CoS templates that compete with each other to provide an attribute value. For example, you might have a multi-valued `cosSpecifier` in your CoS definition entry. In such a case, you can specify a template priority on each template entry to determine which template provides the attribute value. Set the template priority using the `cosPriority` attribute. This attribute represents the global priority of a particular template. A priority of zero is the highest priority.

Templates that contain no `cosPriority` attribute are considered the lowest priority. In the case where two or more templates are considered to supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily.

For example, a CoS template entry for generating a department number appears as follows:

```
dn: cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
departmentNumber: 71776
cosPriority: 0
```

This template entry contains the value for the `departmentNumber` attribute. It has a priority of zero, meaning this template takes precedene over any other conflicting templates that define a different `departmentNumber` value.

The following sections provide examples of template entries along with examples of each type of CoS definition entry.

Example of a Pointer CoS

You want to create a pointer CoS that shares a common postal code with all entries in the `dc=siroe,dc=com` tree.

To add a new pointer CoS definition entry to the `dc=siroe,dc=com` suffix, you do an `ldapmodify` as follows:

```
ldapmodify -a -D "cn=directory manager" -w secret -h host -p 389
```

The `ldapmodify` utility binds to the server and prepares it to add information to the configuration file.

Next, you add the pointer CoS definition to the `dc=siroe,dc=com` root suffix as follows:

```
dn: cn=pointerCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=siroeUS,cn=data,dc=siroe,dc=com
cosAttribute: postalCode
```

Next, you create the template entry as follows:

```
dn: cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438
```

The CoS template entry (`cn=siroeUS,dn=cata,dc=siroe,dc=com`) supplies the value stored in its `postalCode` attribute to any entries located under the `dc=siroe,dc=com` suffix. These entries are the target entries.

Example of an Indirect CoS

This indirect CoS uses the `team` attribute of the target entry to identify the CoS template entry.

First, you add a new indirect CoS definition entry to the `dc=siroe,dc=com` suffix, using `ldapmodify` as follows:

```
ldapmodify -a -D "cn=directory manager" -w secret -h host -p 389
```

The `ldapmodify` utility binds to the server and prepares it to add information to the configuration file.

Next, you add the indirect CoS definition to the `dc=siroe,dc=com` root suffix as follows:

```
dn: cn=indirectCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier: manager
cosAttribute: departmentNumber
```

Next, you create the template entry for the manager Carla Fuentes as follows:

```
dn:cn=Carla Fuentes,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
departmentNumber: 318842
```

You create a second template entry for the manager Sue Jacobs as follows:

```
dn:cn=Sue Jacobs,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
departmentNumber: 71776
```

The definition entry looks in the target entries (the entries under `dc=siroe,dc=com`) for entries containing the `manager` attribute (because this attribute is specified in the `cosIndirectSpecifier` attribute of the definition entry). The `manager` attribute of the target entry can point to one of two templates, `cn=Carla Fuentes,cn=data,dc=siroe,dc=com` and `cn=Sue Jacobs,cn=data,dc=siroe,dc=com`. The department number is different depending upon the manager.

Example of a Classic CoS

You want to create a classic CoS that automatically generates postal codes using a combination of the template DN and the attribute specified in the `cosSpecifier` attribute.

First, you add a new classic CoS definition entry to the `dc=siroe,dc=com` suffix, using `ldapmodify` as follows:

```
ldapmodify -a -D "cn=directory manager" -w secret -h host -p 389
```

The `ldapmodify` utility binds to the server and prepares it to add information to the configuration file.

Next, you add the indirect CoS definition to the `dc=siroe,dc=com` root suffix as follows:

```
dn: cn=classicCoS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=siroeUS,cn=data,dc=siroe,dc=com
cosSpecifier: employeeType
cosAttribute: postalCode override
```

Next, you create the template entries for the sales and marketing departments as follows:

```
dn: cn=sales,cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438

dn: cn=marketing,cn=siroeUS,cn=data,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 99111
```

The classic CoS definition entry applies to all entries under the `dc=siroe,dc=com` suffix. Depending upon the combination of `employeeType` attribute found in the entry and the cosTemplate DN, it can arrive at one of two templates. One, the sales template, provides a postal code specific to employees in the sales department. The marketing template provides a postal code specific to employees in the marketing department.

Creating Role-Based Attributes

You can create classic CoS schemes that generate attribute values for an entry based on the role possessed by the entry. For example, you could use role-based attributes to set the server look through limit on an entry-by-entry basis.

To create a role-based attribute, use the `nsRole` attribute as the `cosSpecifier` in the CoS definition entry of a classic CoS. Because the `nsRole` attribute can be multivalued, you can define CoS schemes that have more than one possible template entry. To resolve the ambiguity of which template entry to use, you can include the `cosPriority` attribute in your CoS template entry.

For example, you can create a CoS that allows members of the manager role to exceed the standard mailbox quota. The manager role exists as follows:

```
dn: cn=ManagerRole,ou=people,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: ManagerRole
nsRoleFilter: o=managers
Description: filtered role for managers
```

The classic CoS definition entry would look as follows:

```
dn: cn=managerCOS,dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectlass: cosClassicDefinition
cosTemplateDn: cn=managerCOS,dc=siroe,dc=com
cosSpecifier: nsRole
cosAttribute: mailboxquota override
```

The `cosTemplateDn` attribute provides a value that, in combination with the attribute specified in the `cosSpecifier` attribute (in the example, the `nsRole` attribute of the target entry), identifies the CoS template entry. The CoS template entry provides the value for the `mailboxquota` attribute. An additional qualifier of `override` tells the CoS to override any existing `mailboxquota` attributes values in the target entry.

The corresponding CoS template entry looks as follows:

```
dn:cn="cn=ManagerRole,ou=people,dc=siroe,dc=com",cn=managerCOS,
 dc=siroe,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectlass: cosTemplate
mailboxquota: 1000000
```

The template provides the value for the `mailboxquota` attribute, 1000000.

NOTE The role entry and the CoS definition and template entries should be located at the same level in the directory tree.

Access Control and CoS

The server controls access to attributes generated by a CoS in exactly the same way as regular stored attributes. However, access control rules depending upon the value of attributes generated by CoS will not work.

Chapter 6

Managing Access Control

iPlanet Directory Server provides you with the ability to control access to your directory. This chapter describes the access control mechanism.

This section includes the following topics:

- Access Control Principles
- Default ACIs
- Creating ACIs Manually
- Bind Rules
- Creating ACIs From the Console
- Access Control Usage Examples
- Viewing the ACIs for an Entry
- Advanced Access Control: Using Macro ACIs
- Access Control and Replication
- Logging Access Control Information
- Compatibility with Earlier Releases

To take full advantage of the power and flexiblity of the access control mechanism, while you are in the planning phase for your directory deployment, you should define an access control strategy as an integral part of your overall security policy. Refer to *iPlanet Directory Server Deployment Guide* for tips on planning your access control strategy.

Access Control Principles

The mechanism by which you define access is called *access control*. When the server receives a request, it uses the authentication information provided by the user in the bind operation, and the access control instructions (ACIs) defined in the server to allow or deny access to directory information. The server can allow or deny permissions such as read, write, search, and compare. The permission level granted to a user may be dependent on the authentication information provided.

Using access control, you can control access to the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes. You can set permissions for a specific user, all users belonging to a specific group or role, or all users of the directory. Finally, you can define access for a specific location such as an IP address or a DNS name.

ACI Structure

Access control instructions are stored in the directory, as attributes of entries. The `aci` attribute is an operational attribute; it is available for use on every entry in the directory, regardless of whether it is defined for the object class of the entry. It is used by the directory server to evaluate what rights are granted or denied when it receives an LDAP request from a client. The `aci` attribute is returned in an `ldapsearch` operation if specifically requested.

The three main parts of an ACI statement are:

- Target
- Permission
- Bind Rule

The permission and bind rule portions of the ACI are set as a pair, also called an Access Control Rule (ACR). The specified permission is granted or denied depending on whether the accompanying rule is evaluated to be true.

ACI Placement

If an entry containing an ACI does not have any child entries, the ACI applies to that entry only. If the entry has child entries, the ACI applies to the entry itself and all entries below it. As a direct consequence, when the server evaluates access permissions to any given entry, it verifies the ACIs for every entry between the one requested and the directory suffix, as well as the ACIs on the entry itself.

The `aci` attribute is multi-valued, which means that you can define several ACIs for the same entry or subtree.

You can create an ACI on an entry that does not apply directly to that entry but to some or all of the entries in the subtree below it. The advantage of this is that you can place at a high level in the directory tree a general ACI that effectively applies to entries more likely to be located lower in the tree. For example, at the level of an `organizationalUnit` entry or a `locality` entry, you could create an ACI that targets entries that include the `inetorgperson` object class.

You can use this feature to minimize the number of ACIs in the directory tree by placing general rules at high level branch points. To limit the scope of more specific rules, you should place them as close as possible to leaf entries.

NOTE ACIs placed in the root DSE entry apply only to that entry.

ACI Evaluation

To evaluate the access rights to a particular entry, the server compiles a list of the ACIs present on the entry itself and on the parent entries back up to the top level entry stored on the directory server. ACIs are evaluated across all of the databases for a particular directory server, but not across directory servers.

The evaluation of this list of ACIs is done based on the semantics of the ACIs, not on their placement in the directory tree. This means that ACIs that are close to the root of the directory tree do not take precedence over ACIs that are closer to the leaves of the directory tree.

The precedence rule that applies is as follows: ACIs that deny access take precedence over ACIs that allow access. Between ACIs that allow access, union semantics apply, so there is no precedence.

Using Attributes in Search Filters

When searching for an entry, you can specify attributes associated with that type of entry. For example, when you search for people entries, you can use the `cn` attribute to search for people with a specific common name.

Examples of attributes that people entries might include:

- `cn` (the person's common name)
- `sn` (the person's surname, or last name, or family name)
- `telephoneNumber` (the person's telephone number)
- `buildingName` (the name of the building in which the person resides)
- `l` (the locality where you can find the person)

For a listing of the attributes associated with types of entries, see the *iPlanet Directory Server Schema Reference.*

Using Operators in Search Filters

The operators that you can use in search filters are listed in Table B-1:

Table B-1 Search Filter Operators

Search type	Operator	Description
Equality	=	Returns entries containing attribute values that exactly match the specified value. For example, `cn=Bob Johnson`

Substring	*=string* string*	Returns entries containing attributes containing the specified substring. For example,
		`cn=Bob*`
		`cn=*Johnson`
		`cn=*John*`
		`cn=B*John`
		(The asterisk (*) indicates zero (0) or more characters.)

Table B-1 Search Filter Operators *(Continued)*

Search type	Operator	Description
Greater than or equal to	>=	Returns entries containing attributes that are greater than or equal to the specified value. For example, `buildingname >= alpha`
Less than or equal to	<=	Returns entries containing attributes that are less than or equal to the specified value. For example, `buildingname <= alpha`

Presence	=*	Returns entries containing one or more values for the specified attribute. For example,
		`cn=*`
		`telephonenumber=*`
		`manager=*`

Approximate	~=	Returns entries containing the specified attribute with a value that is approximately equal to the value specified in the search filter. For example,
		`cn~=suret`
		`l~=san fransico`
		could return
		`cn=sarette`
		`l=san francisco`

NOTE In addition to these search filters, you can specify special filters to work with a preferred language collation order. For information on how to search a directory with international character sets, see "Searching an Internationalized Directory," on page 497.

Using Compound Search Filters

Multiple search filter components can be combined using Boolean operators expressed in prefix notation as follows:

(*Boolean-operator* (*filter*) (*filter*) (*filter*) . . .)

where *Boolean-operator* is any one of the Boolean operators listed in Table B-2.

Boolean operators can be combined and nested together to form complex expressions, such as:

(*Boolean-operator* (*filter*) ((*Boolean-operator* (*filter*) (*filter*)))

The Boolean operators available for use with search filters include the following:

Table B-2 Search Filter Boolean Operators

Operator	Symbol	Description
AND	&	All specified filters must be true for the statement to be true. For example, `(&(filter)(filter)(filter)...)`
OR	|	At least one specified filter must be true for the statement to be true. For example, `(|(filter)(filter)(filter)...)`
NOT	!	The specified statement must not be true for the statement to be true. Only one filter is affected by the NOT operator. For example, `(!(filter))`

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first
- All expressions from left to right

Search Filter Examples

The following filter searches for entries containing one or more values for the manager attribute. This is also known as a presence search:

```
manager=*
```